北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

软件理论与工程

需求工程

主讲：高广宇

讲授内容

- 软件需求
- 需求工程过程
- 需求建模
- 形式化描述

1. 什么是需求？

- **需求**是对系统应该提供的服务和所受约束的描述。
- 由于需求要向不同类型的涉众（读者）传达不同层次的信息，可以将需求分为：
  - **用户需求(目标需求)**：用用户所熟悉的表达形式给出需求描述。
  - **系统需求(产品需求)**：详细地给出系统将提供的服务以及系统所受到的约束，比用户需求更具体，更形式化。
  - **软件设计描述(设计层需求)**：在系统需求描述的基础上再加入更加详细的设计层面的需求细节。

示例1

用户需求

1. 软件必须能够访问外部文件，这些外部文件是由其它工具创建的
2. ……

系统需求

1.1 为用户提供定义外部文件类型的工具。
1.2 每种外部文件类型在界面上用一种专门的图标来表示 。
1.3 当用户选择一个代表外部文件的图标时，与该外部文件类型相关联的工具启动。
1.4 ……
2.1 …

示例2

R1. 预算误差<5%	目标需求
R2. 支持报价注册、更新，以及根据需求随时调整报价	业务需求
R3. 产品应具有记录、检索历史报价的功能	产品需求
R4.系统界面大致如附件xx所示	目标需求

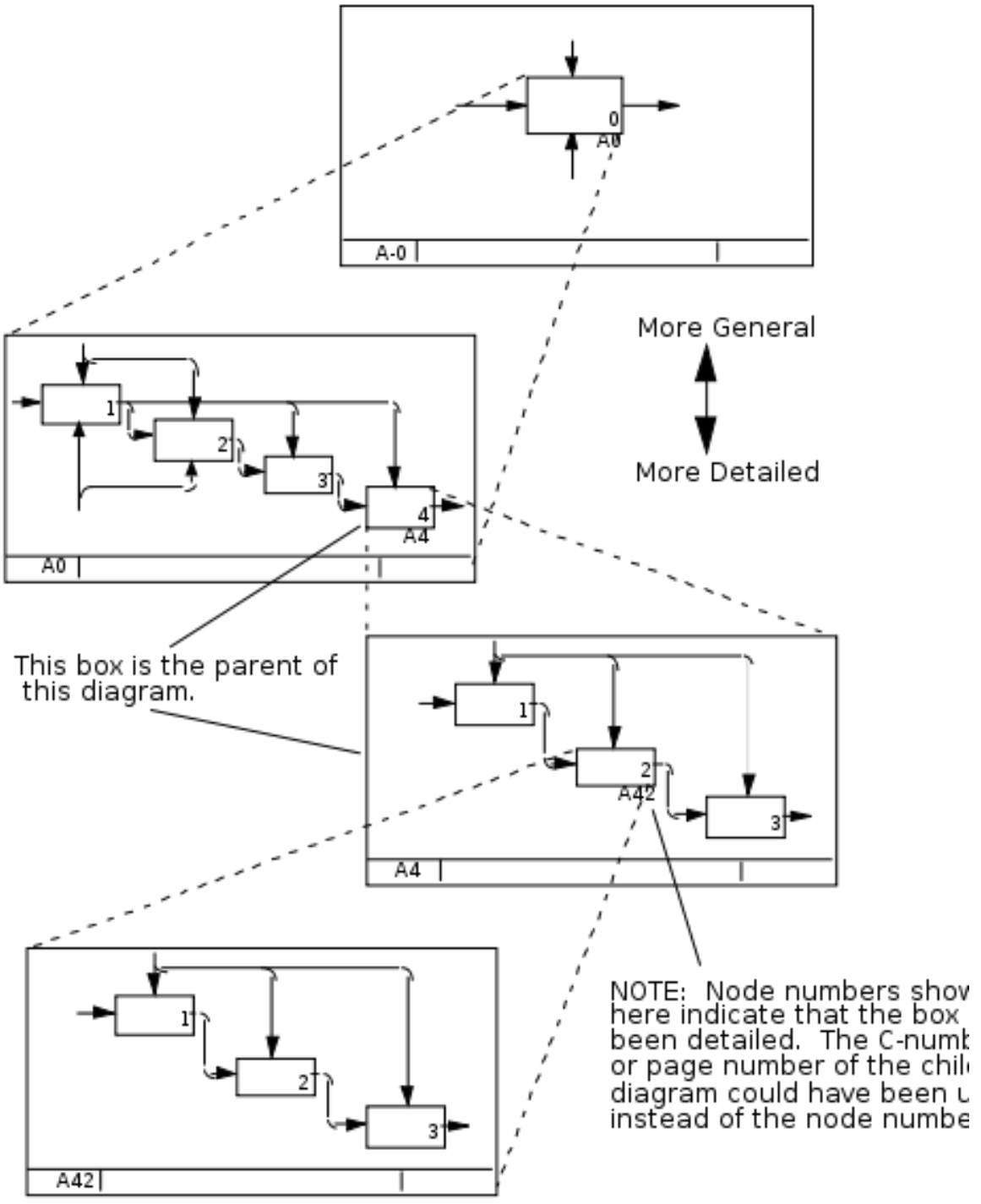
- 用自然语言描述的用户需求
  - **描述不够清楚**（二义性）
  - **需求混乱**（功能需求、非功能需求、系统目标和设计信息无法清晰地区分）
  - **需求混合**（多个不同的需求交织在一起，以一个需求的形式给出）
- 描述系统需求可能用到多种不同模型，如：对象模型、数据流模型等

- 原则上讲，系统需求仅仅描述做什么，而不应该描述如何实现。然而，要给出细节需求而不提到任何设计信息，事实上也是不可能的：
  - 通常系统需求依照构成系统的各个子系统结构来给出，即由初始的系统体系结构来构造需求描述；
  - 通常目标系统和已有系统互操作，这就约束了目标系统的设计，同时这些约束又构成了新系统的需求；
- 某些特别的设计（如NVP）是系统的一个外部需求

系统需求描述工具

描述工具	说明
结构化自然语言	依赖于定义标准格式或模板来表达需求
PDL语言	比一般的计算机高级语言更接近自然语言
图形化工具	通过图形语言（辅之于文本注释）来定义系统的功能需求。如SADT,基于用例的描述等
形式化工具	基于有限状态机、集合等数学工具形式化地描述需求。

SADT: Structured Analysis and Design Techniques



2. 需求的另一种划分

- 业务需求
- 用户需求
- 功能需求
- 非功能需求

- **业务需求**（Business Requirement）
  - 反映了组织机构或客户对系统、产品的高层次目标要求
  - 反映目标系统所处领域的特点
  - 在项目视图与范围文档中予以说明

- **用户需求**（User Requirement）
  - 用户使用产品必须要完成的任务
  - 在使用用例文档或方案脚本说明中予以说明

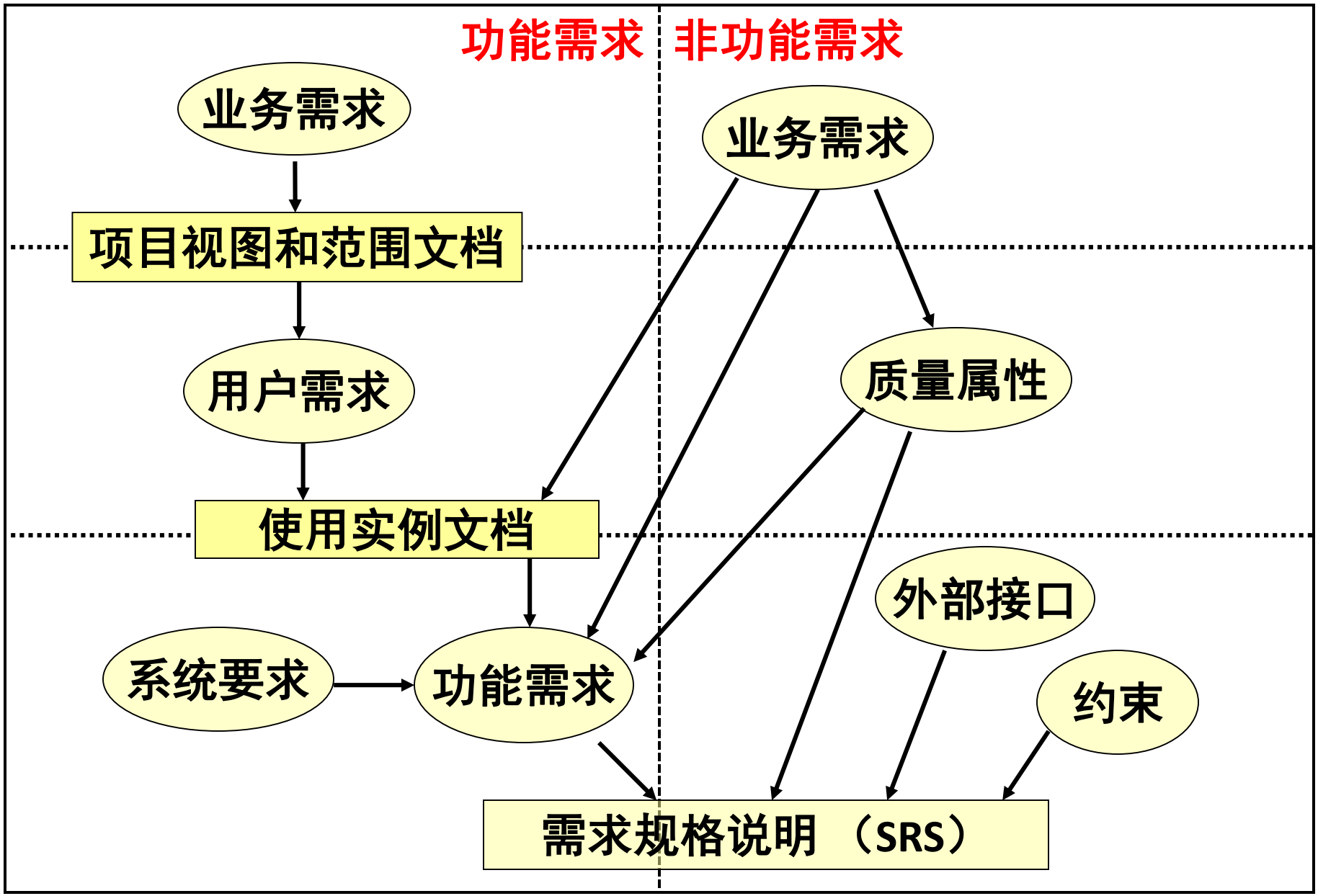
- **功能需求**（Functional, Behavioral Requirement ）
  - 定义了开发人员必须实现的软件功能，使得用户能完成他们的任务，从而满足了业务需求
  - 对系统应该提供的服务、如何对输入做出发应以及系统在特定条件下的行为的描述。
  - 涉及与本系统有接口的其他系统的所有事情。
  - 可能需要明确声明系统不应该做什么。

- **非功能需求(Non-functional Requirement)**
  - 对系统提供的服务或功能给出的约束，包括性能指标、对质量属性（quality attribute）的描述、外部接口以及设计与实现的约束（constraint）、时间约束、标准等。
  - 非功能需求最好是可以验证的，但实际上对需求量化通常很难。
  - 非功能需求与功能需求有时会发生冲突。
  - 非功能需求之间会发生冲突。

非功能需求分类

目标系统的限制	性能	实时性、精确度、资源利用率等
	可靠性	
	安全/保密性	
	运行限制	使用频度、运行期限、控制方式、操作要求
	物理限制	系统规模等限制
开发维护的限制	开发类型	实用性开发、试验性开发
	开发工作量的估计	
	开发方法	质量控制标准、里程碑和评审、验收标准
	优先性和可修改性	
	可维护性	

各类需求之间的关系


功能需求
非功能需求
业务需求
项目视图和范围文档
用户需求
使用实例文档
系统要求
功能需求
业务需求
质量属性
外部接口
约束
需求规格说明 （SRS）

功能需求示例：大学图书馆系统

1. 用户能从数据库中查询，或选择一个子集查询。
2. 系统能够提供**多种**浏览器供用户阅读馆藏文献。
3. 每次借阅能够对应一个唯一的识别符，可拷贝到用户的常备存储区内。

- 功能需求以不同的详细程度重写（需求1和3）
- 含糊的表达，“**多种**浏览器”

容易忽略的非琐碎要求（示例）

酒店房间预订系统：

R1：根据客房类型而不是客房号进行预订（业务细节）

R2：考虑到预订客房的客户有可能不入住，可以接受超过空闲客房数量的预订 （边界条件）

R3：授权的系统管理员可以自定义单价（权限）

3.软件需求文档

- SRS (Software Requirement Specification)
- Heninger 对软件需求文档提出的6点要求：
  - It should specify external system behavior.
  - It should specify constraints on the implementation.
  - It should be easy to change.
  - It should serve as a reference tool for system maintainers.
  - It should record forethought about the life cycle of the system.
  - It should characterize acceptable responses to undesired events.

北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

SRS结构

- Introduction
  - Purpose
  - Definitions
  - System overview
  - References
- Overall description
  - Product perspective
    - System Interfaces
    - User Interfaces
    - Hardware interfaces
    - Software interfaces
    - Communication Interfaces
    - Memory Constraints
    - Operations
    - Site Adaptation Requirements
  - Product functions
  - User characteristics
  - Constraints, assumptions and dependencies
- Specific requirements
  - External interface requirements
  - Functional requirements
  - Performance requirements
  - Design constraints
    - Standards Compliance
  - Logical database requirement
  - Software System attributes
    - Reliability
    - Availability
    - Security
    - Maintainability
    - Portability
  - Other requirements

北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

SRS撰写要求

- 正确性
- 无二义性（需求确实是用户所需吗？）
- 完整性（完备性，包括用户需要的每一功能或性能）
- 一致性（需求之间不能互相矛盾）
- 可检验性（非计算机人员可以理解）
- 可实现性（有效性，需求是能够现实的吗？需要什么硬件系统支持？）
- 可修改性
- 可跟踪性
- 注释

讲授内容

- 软件需求
- 需求工程
- 需求建模
- 形式化描述

1. 什么是需求工程？

- **需求工程**是对服务和约束的发现、分析、描述和检验的过程。
- 需求工程的4个高层通用过程：
  - 可行性研究
  - 需求导出和分析
  - 需求描述和文档编写
  - 需求有效性验证

需求分析的涉众

- 合同监管人员(PM)：提出里程碑(Milestones)和约束系统开发进度的计划
- 需求者：客户(Customer)和使用者(User)。
- 开发者
  - **系统分析人员**
  - **设计人员**，依据需求提出可接受的解决方案。
  - **测试人员**，确保软件系统满足每一需求。
  - **系统集成人员**

系统分析人员应具有的素质

- 综合能力
  - 总体规划，抽象和分解能力
- 保证整个过程的善始善终的能力
- 交流、理解和表达能力
- 技术水平
  - 了解问题域和描述解空间的能力

重视需求分析

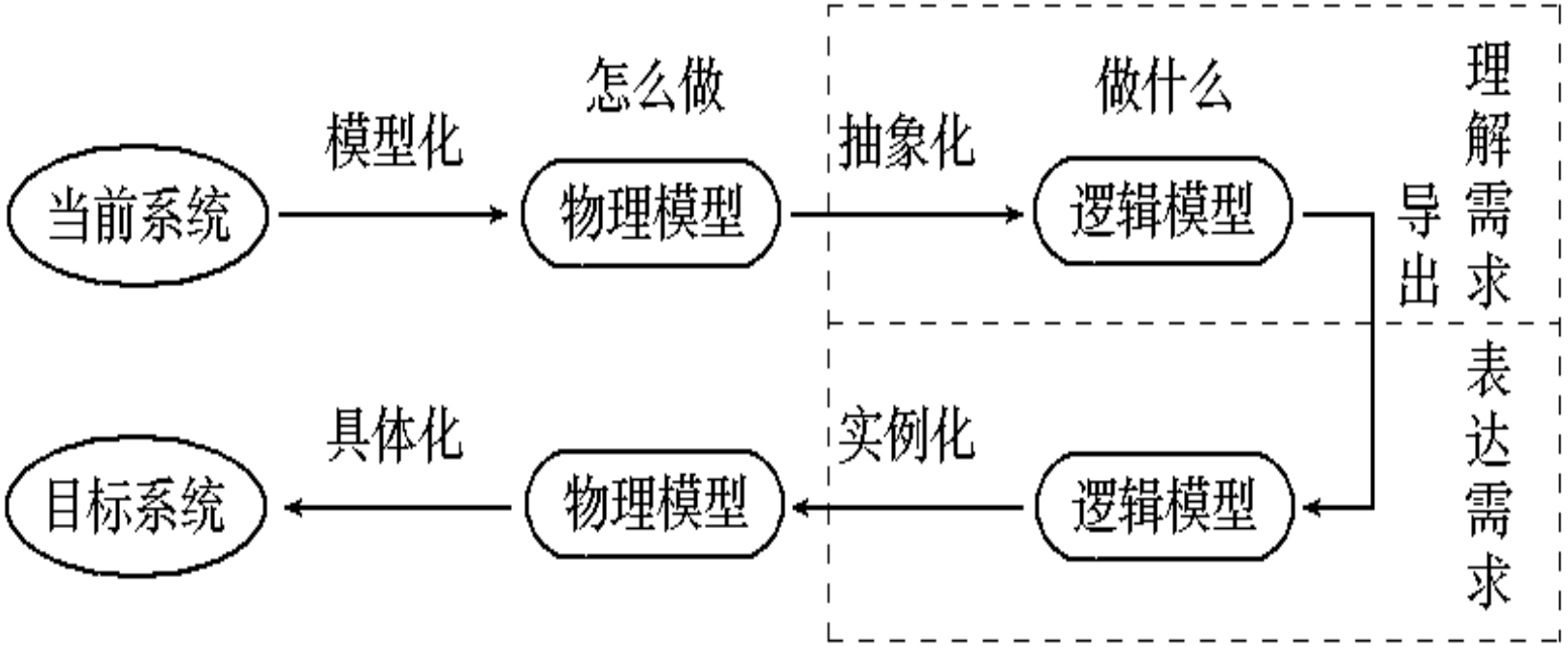
- 软件项目中40%—60%的缺陷都是由需求分析阶段的过失所致（Davis 1993，Leffingwell 1997）
- 对欧洲软件行业的大规模调查显示：确定和管理用户需求是问题最多的两个环节（ESPITI 1995）
- 许多软件问题都源于收集、记录、协商和修改产品需求过程中的方式不当
  - 信息收集方式不正规
  - 没有明确提出想要的功能
  - 常常存在未经沟通的错误假设
  - 需求的定义不够充分
  - 未经仔细考虑进行需求变更
  - ……

2. 可行性研究

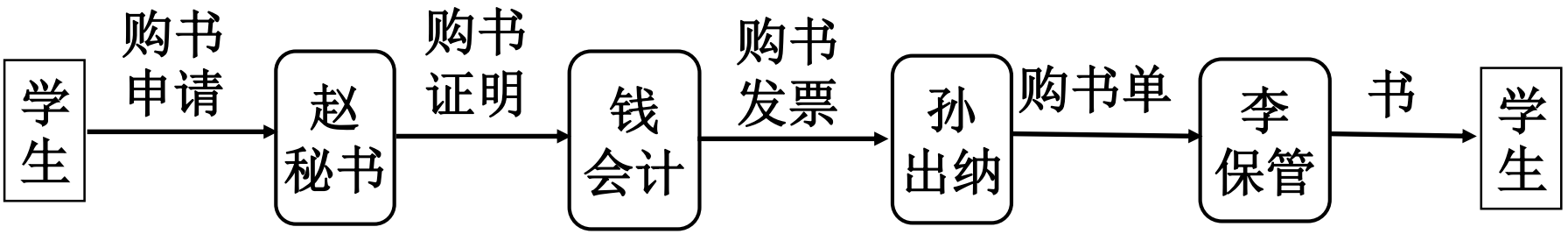
- 焦点问题：
  - 系统是否符合总体目标？
  - 系统是否能在现有条件和预算内按时完成？
  - 目标系统能与已有系统集成？

- 技术可行性
- 经济可行性
- 社会可行性
- 操作可行性

可行性研究是高层的分析和设计

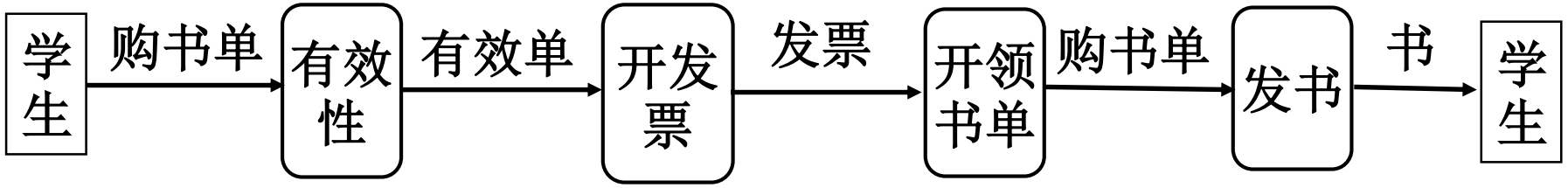


学生在学校教材科购买教材的系统的例子



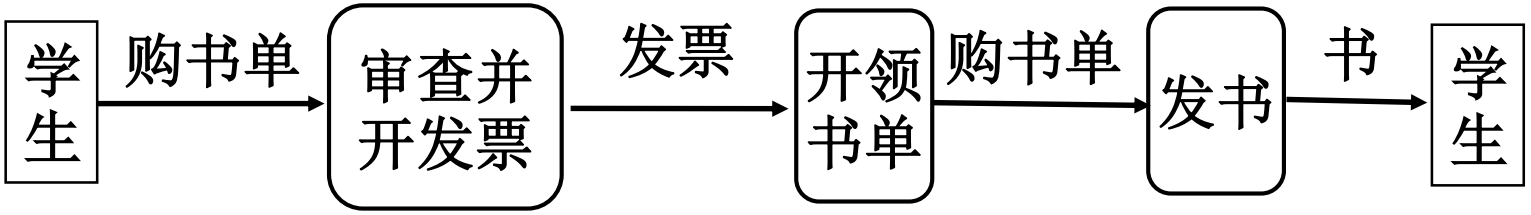
一、 通过对现实环境的研究，获得系统具体物理模型

学生在学校教材科购买教材的系统的例子



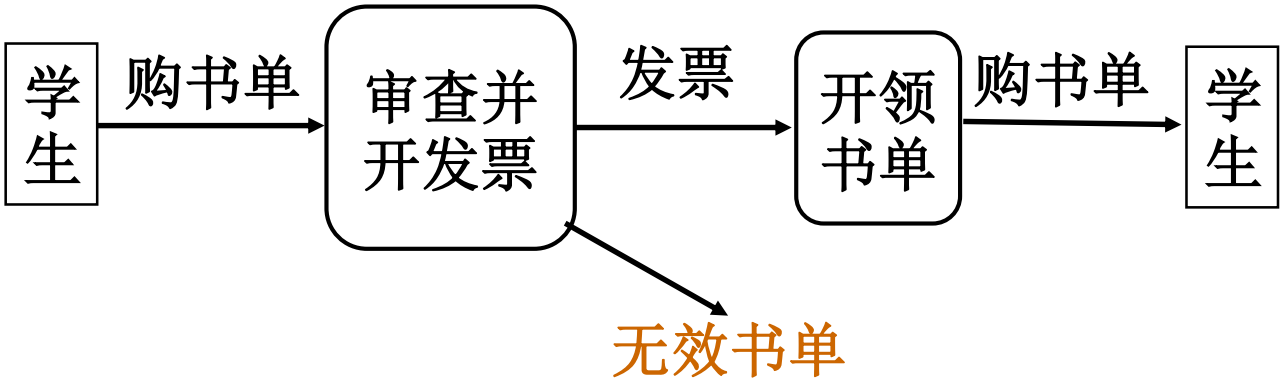
二、 去掉具体模型的非本质因素，抽取出逻辑模型

学生在学校教材科购买教材的系统的例子



三、 分析当前系统和目标系统的差别，
建立目标系统的逻辑模型。

学生在学校教材科购买教材的系统的例子



四、 对目标系统进行完善和补充，写出完整的需求说明。

五、 对需求说明进行复审，直到确认文档齐全并符合用户的全部需求为止。

北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

可行性研究报告

（1）引言
（2）可行性研究前提
（3）对现有系统的分析
（4）所建议系统的技术可行性分析
（5）所建议系统的经济可行性分析
（6）社会因素可行性分析
（7）其它可供选择方案
（8）结论意见

需求与什么相关？

1) 物理环境(Physical Environment)
2) 接口(Interfaces)
3) 用户或人的因素(Factors)
4) 功能(Functionality)
5) 文档(Documentation)
6) 数据(Data)
7) 资源(Resources)
8) 安全性(Security)
9) 质量保证(Quality Assurance)

1）物理环境 (Physical Environment)

- 设备的主要用途，在哪里发挥什么作用？
- 所须设置设备的多少？
- 环境限制等，如温度、湿度或干扰？

2） 接口描述（Interface Description）

- 来自一个或多个其他系统的输入？
- 对一个或多个其它系统的输出？
- 数据是否必须预先进行规定的格式化处理？
- 数据是否需要预先存放的介质？

3） 用户和人为因素

- 谁使用系统？
- 有几种类型的用户？
- 每种类型用户的技术水平怎样？
- 对每型用户需要什么样的培训？
- 用户理解、使用系统的难易度怎样？
- 用户误用系统的困难程度怎样？

4）功能描述（Function Description）

- 系统将做什么？
- 系统将在何时做？
- 有几种操作方式？
- 系统能在何时、怎样被改变或增强？
- 对执行速度，响应时间或数据流量有何限制和约束？

5） 文档（Documents）

- 需要多少文档？
- 是联机文档还是静态文档或者二者皆可？
- 文档所面向的对象（读者）？

6）数据（Data）

- I/O数据格式应该是什么样的？
- 数据收或发的频度？
- 数据的精确度
- 数据流量？
- 数据必须在何时予以保存，保存多久？

7） 资源描述（Resource Description）

- 建立和维护系统都要什么材料、人员或其他资源？
- 开发者必须具有哪些技术？
- 系统占用空间？
- 开发时间表？
- 资金限制？

8）安全性描述 (Security)

- 对系统或信息的存取必须在我们的控制之下？
- 不同用户的数据之间将如何实现隔离？
- 不同用户程序之间，以及和操作系统间怎样隔离？
- 系统如何备份？
- 备份副本必须被存于一个不同的位置？

- 物理安全：应采取措施防火，防水防盗等安全措施？

9）质量保证 (Quality Assurance)

- 系统必须有效检测并隔离故障？
- 平均无故障时间规定为多少？
- 对一次失败后重启系统有一个最大时间？
- 系统如何将变化合并到设计？
- 维护仅仅是纠正错误，还是包括改进系统？
- 对资源和响应时间使用什么样的有效度量？
- 系统移植性、可维护性等要求？
- 如何向别人示范系统的特征？

4. 需求验证

- 验证是为了确保需求说明准确、完整地表达必要的质量特点
- 审查需求文档
- 以需求为依据编写测试用例
  - 写出黑盒功能测试用例
- 编写用户手册
  - 要用浅显易懂的语言描述出所有对用户可见的功能
- 确定合格的标准

验证的方法

- 复审和进一步需求分析
- 实现原型系统
- 支持需求分析工具

验证的主要内容

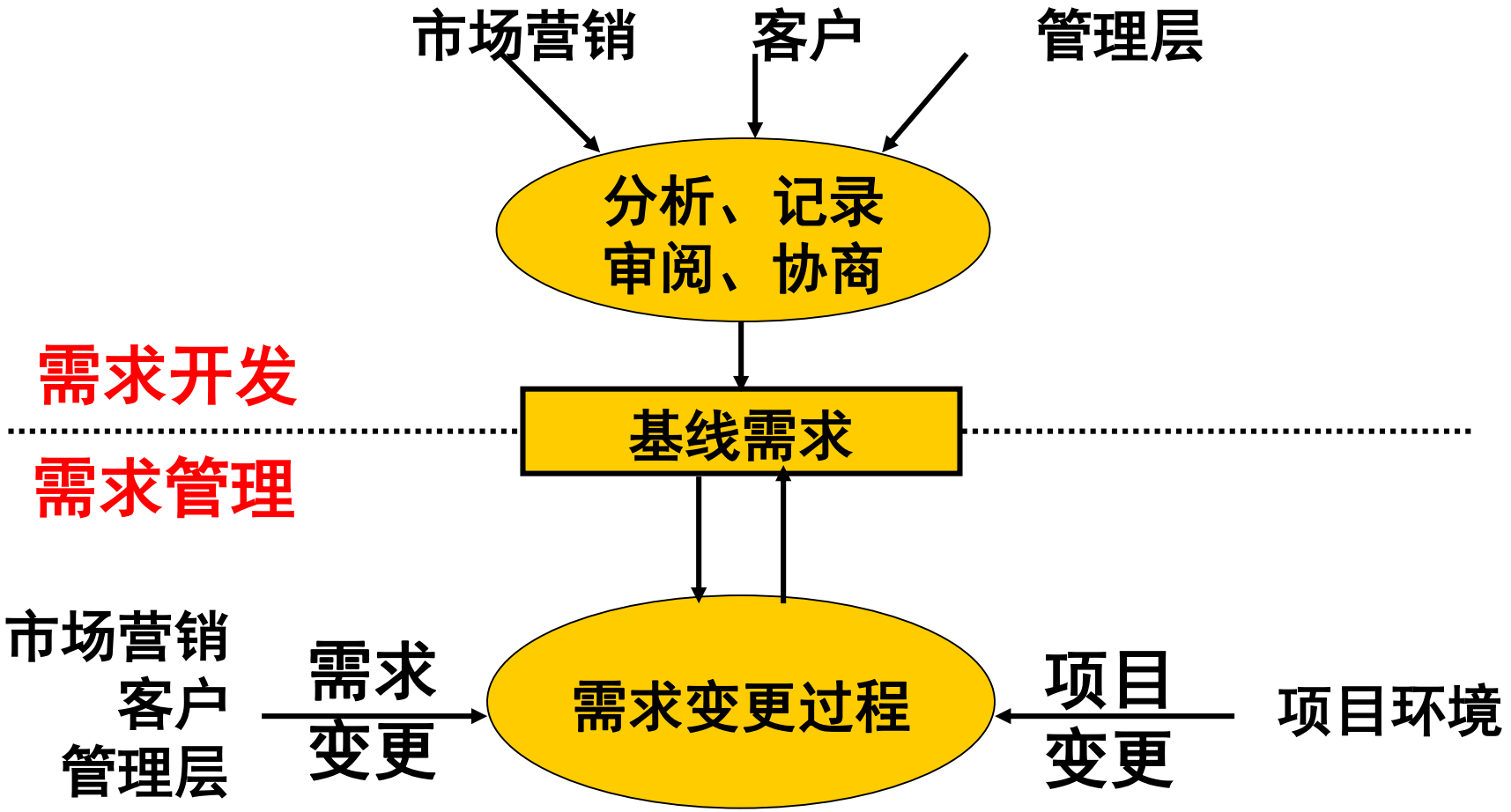
- 一致性验证
- 现实性验证（需求是现实的吗？）
- 完整性（完备性）和有效性验证

5. 需求管理

- 需求管理是对需求变更了解和控制的过程。
- 需求管理的任务是“与客户就软件需求达成并保持一致”（Paulk 1995）
- 持久的需求与易变的需求
- 一个变更管理过程由三个阶段
  - 问题分析和变更描述
  - 变更分析和成本计算
  - 变更实现

需求管理活动

- 定义需求基线
- 审查变更请求，评估可能产生的影响以决定是否批准
- 以可控的方式将批准的变更融入项目中
- 保持项目计划与需求的同步
- 基于对需求变更影响的估计协商新的需求约定
- 跟踪每项需求（找到对应的设计、代码和测试用例）
- 在项目的开发过程中始终跟踪需求的状态和变更


市场营销
客户
管理层
分析、记录
审阅、协商
需求开发
基线需求
需求管理
市场营销
客户
管理层
需求
变更
需求变更过程
项目
变更
项目环境

对项目需求状况作出快速评估（1）

1) 项目前景(vision)和范围(scope)未曾明确定义
2) 客户太忙，没时间与需求分析和开发人员一起讨论需求
3) 用户代理（如开发经理、用户负责人、营销人员等）自诩可以代表用户，其实不能准确说出用户的要求
4) 需求只存在于那些所谓专家的脑子里，没有被记录下来
5) 客户坚持所有需求都很重要，不愿排出它们的优先次序
6) 开发人员在编码过程中发现需求有歧义，缺少足够的信息，只能去猜测

对项目需求状况作出快速评估（2）

7) 开发人员与客户沟通时只关心用户界面，忽略了用户需要用软件做什么
8) 客户签字确认了需求却又一直提出修改要求
9) 项目范围因接受需求变更而扩大，却没有相应地增加投入或剪裁功能，进度因而被延误
10) 需求变更的请求被弄丢，开发人员和客户都不了解所有变更请求的状态
11) 开发人员按照客户要求实现的功能无人问津
12) 需求规格说明中的要求都实现了客户却不满意

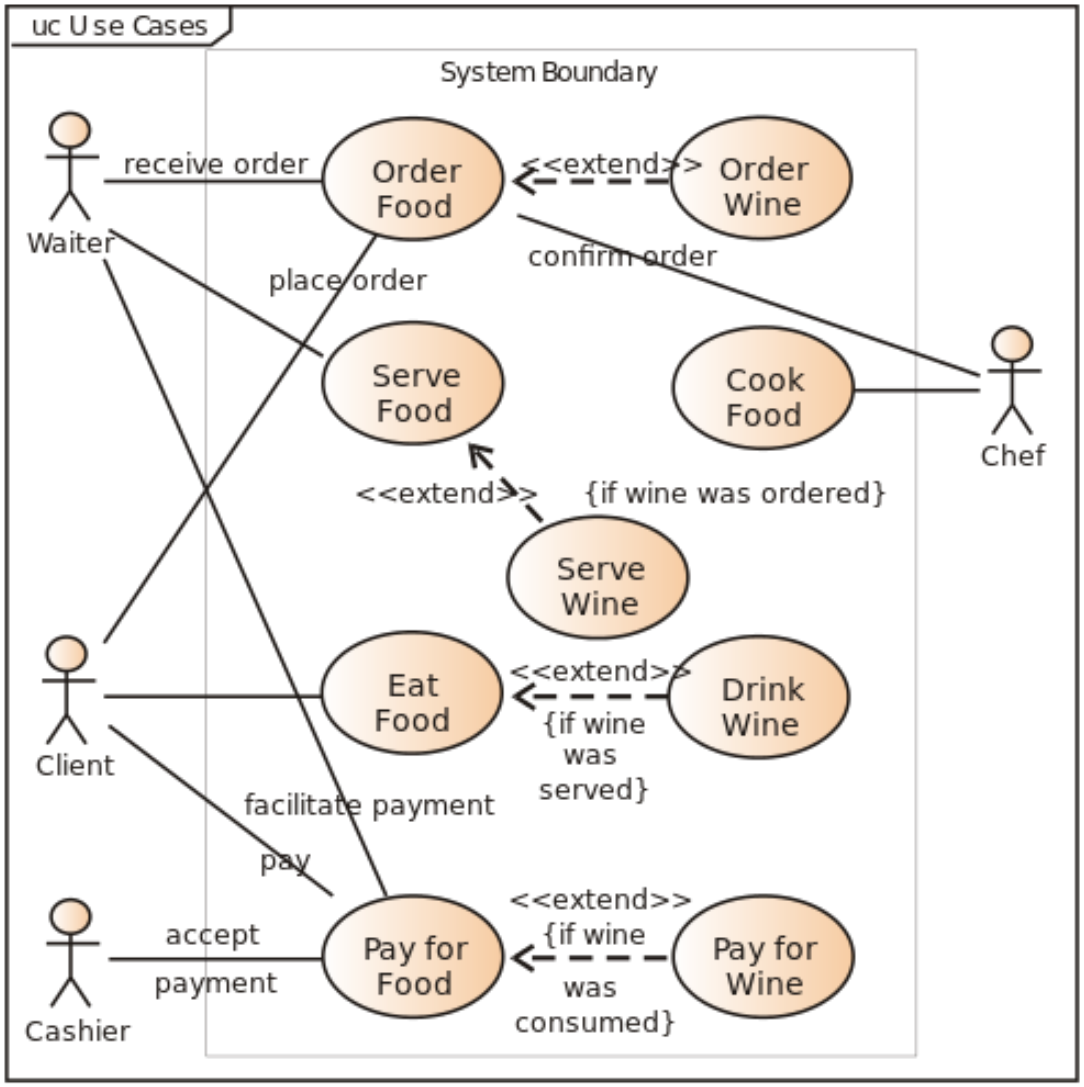
讲授内容

- 软件需求
- 需求工程
- 需求建模
- 形式化描述

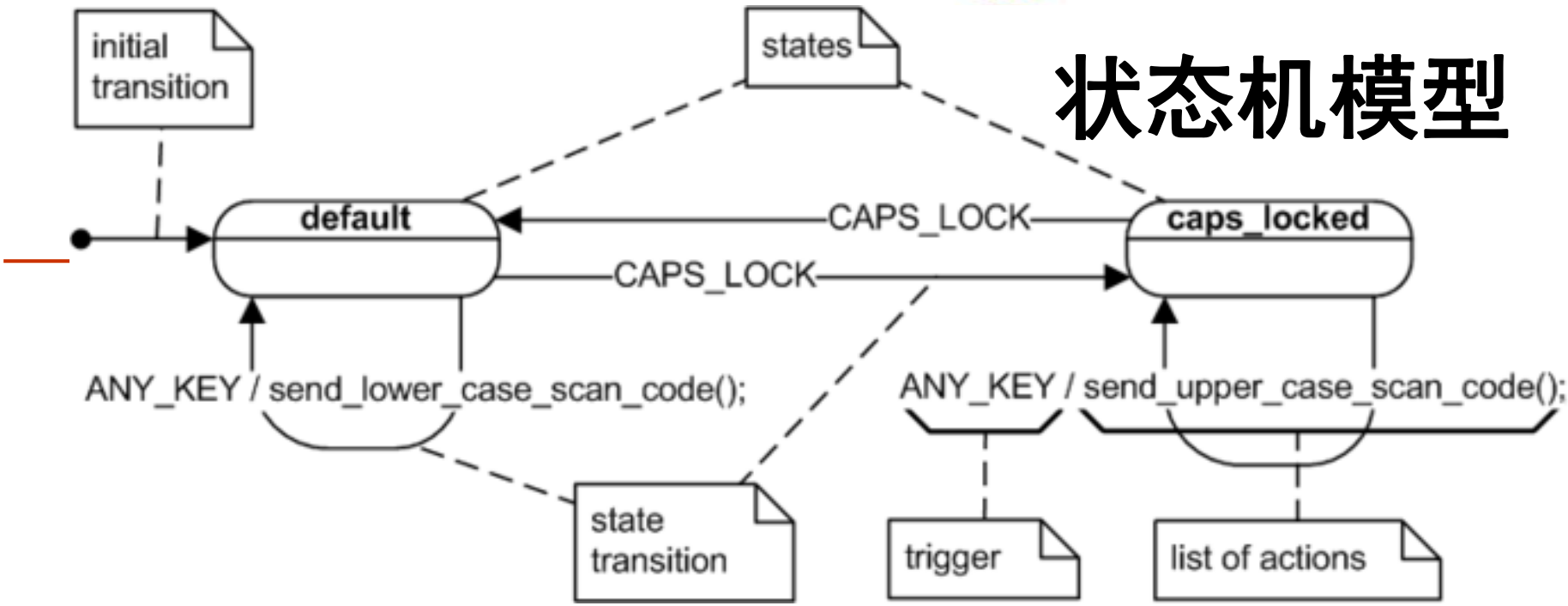
需求模型

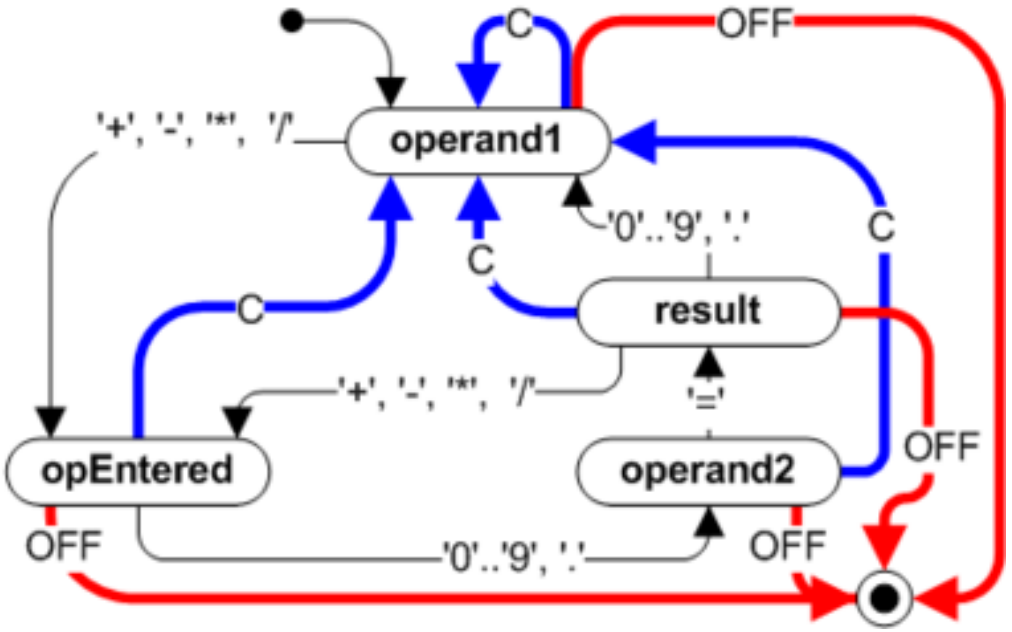
- **模型**是系统的抽象视图，它忽略了系统中的所有细节。
- 从不同的角度（外部、行为或结构）表达系统，形成不同类型的模型：**上下文模型**、**行为模型**、**结构模型**。
- **上下文模型**（系统环境模型）表达系统在整个环境中与其它系统和过程的位置关系。如：用例模型
- **状态机模型**用来描述系统的行为，以响应内部和外部的事件。它是一种**行为模型**。
- **结构模型**包括体系结构模型和数据结构模型。
- **数据流模型**用来描述数据是怎样一步步在处理序列中流动的，它不仅可以描述系统内的处理过程（行为），也能够有效地描述系统的上下文。
- **E-R模型**是一种最广泛使用的**数据结构模型。**
- **对象建模**在一定程度上是结构建模和行为建模的结合。UML已经被OMG认定为对象建模标准。

用例模型


uc Use Cases
System Boundary
Waiter
receive order
Order Food
<<extend>>
Order Wine
confirm order
place order
Serve Food
Cook Food
Chef
<<extend>>
{if wine was ordered}
Serve Wine
Client
Eat Food
<<extend>>
Drink Wine
{if wine was served}
facilitate payment
pay
Cashier
accept payment
Pay for Food
<<extend>>
{if wine was consumed}
Pay for Wine

状态机模型


initial transition
states
default
CAPS_LOCK
caps_locked
CAPS_LOCK
ANY_KEY / send_lower_case_scan_code();
ANY_KEY / send_upper_case_scan_code();
state transition
trigger
list of actions


C
OFF
'+', '-', '*', '/'
operand1
'0'..'9', '.'
C
C
C
result
'+', '-', '*', '/'
'='
opEntered
operand2
OFF
OFF
'0'..'9', '.'
OFF

Entity-Relationship Model

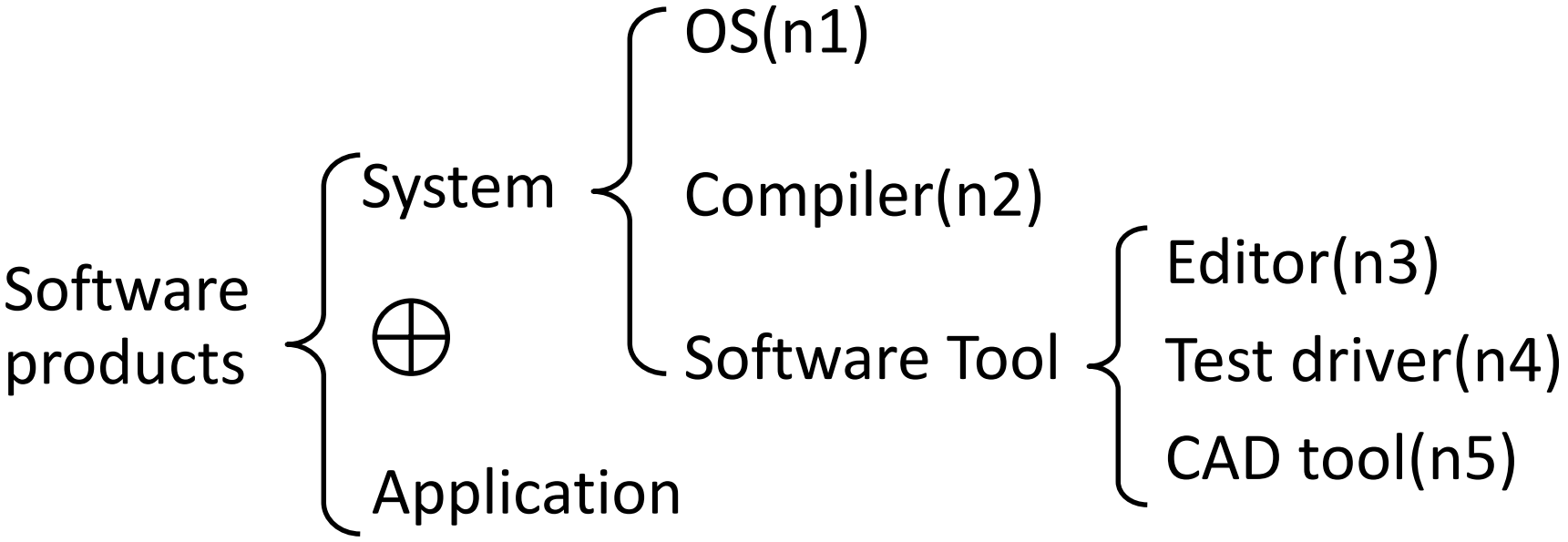
- 概念模型（E-R图）
- 逻辑模型（二维表的定义）
- 物理模型（存储空间的定义，如定义各个字段的大小）
- 数据库的设计一般应经过由概念模型到逻辑模型，再到物理模型的映射过程。

Decision Table

	Rule1	Rule2	Rule3	Rule4	Rule5
High standardized exam scores	T	F	F	F	F
High grades	—	T	F	F	F
Outside activities	—	—	T	F	F
Good recommendations	—	—	—	T	F
Send rejection letter			X	X	X
Send admission forms	X	X			

北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

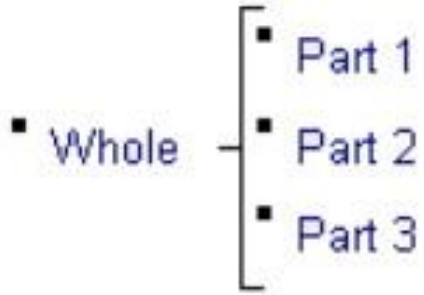
Warnier-Orr Diagrams



8 fundamental building blocks

Hierarchy

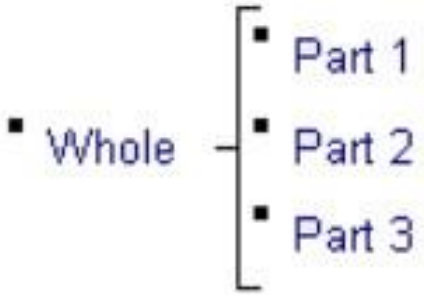
The hierarchy operations breaks things into parts.



The whole *consists of* Part 1 and Part 2 and Part 3.

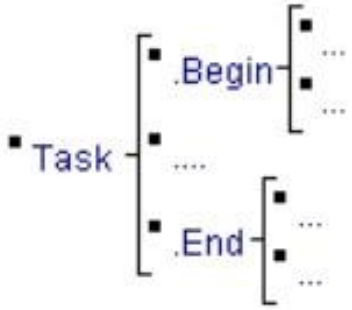
Complement

Complement is the logical NOT.
The Action Code is either Valid or NOT.



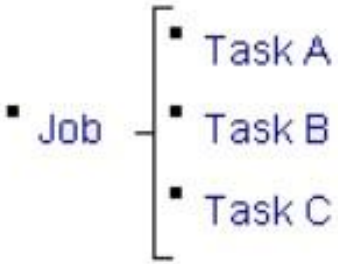
Begin/End Blocks

The Begin block performs initialization and the End block performs termination.



Sequence

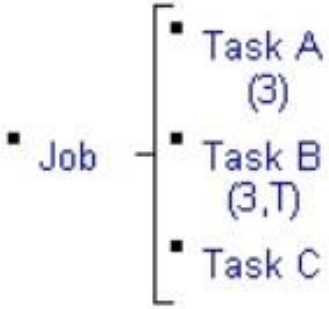
Sequence orders things.



Job *consists of* first Task A followed by Task B, then Task C.

Repetition

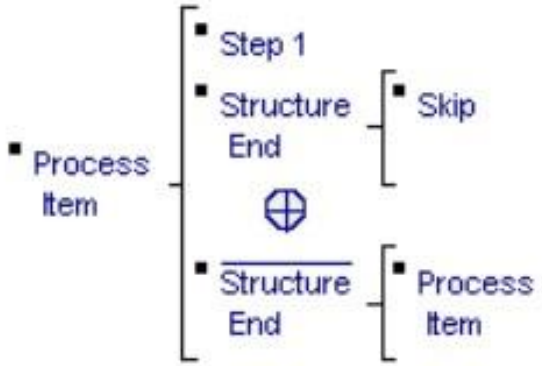
Repetition provides looping.



Job *consists of* first doing Task A 3 times followed by 3 to T repetitions of task B, then doing Task C one time.

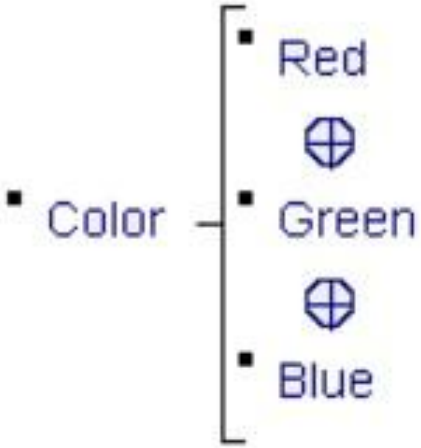
Recursion

A recursive process contains itself as a sub-process.



Selection

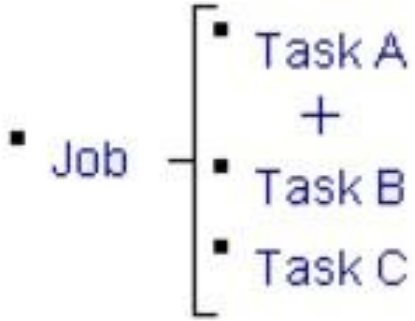
Selection allows choices.



Color *consists of* either red **or** green **or** blue.

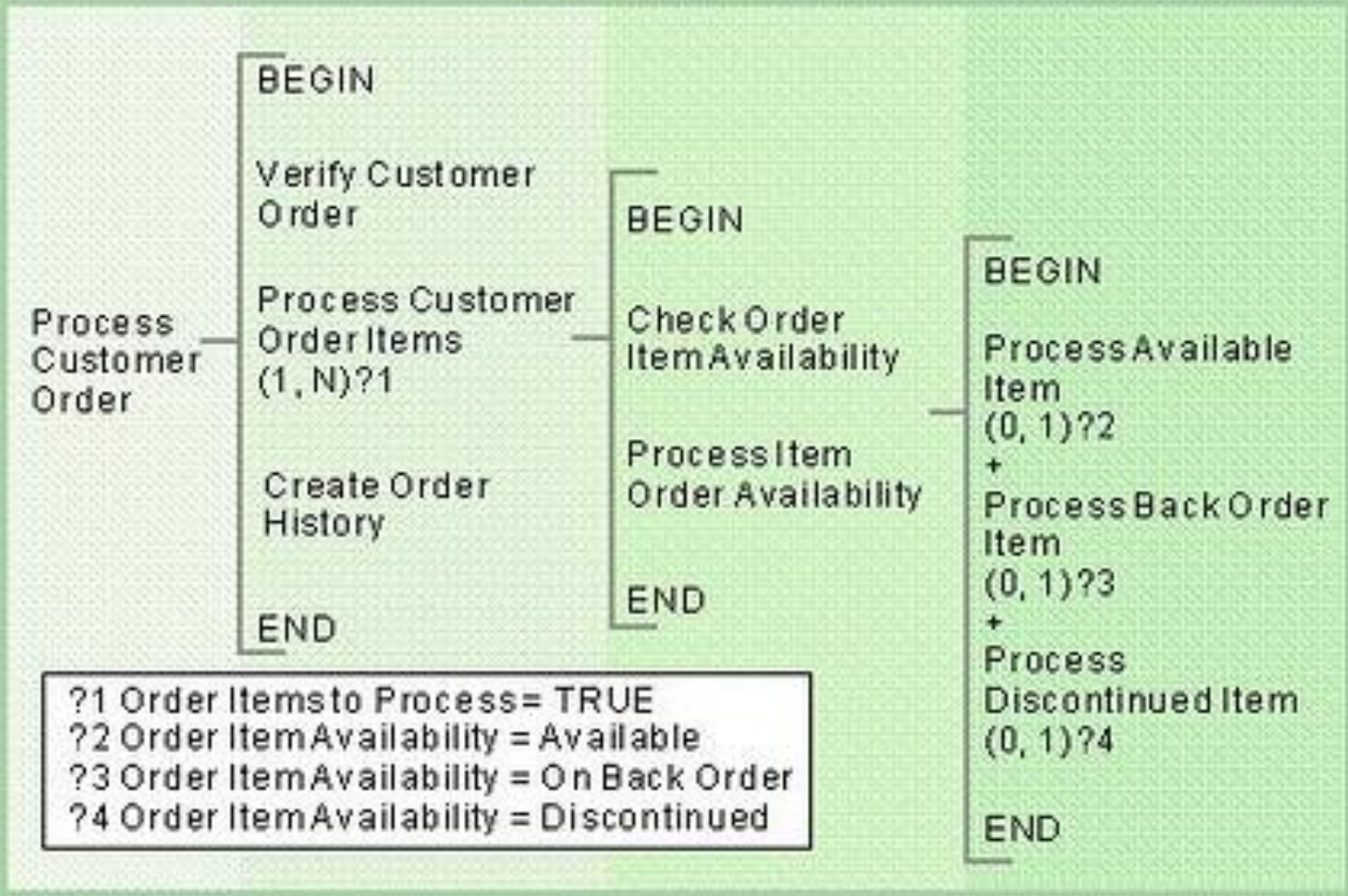
Concurrency

Concurrency allows things to happen at the same time.



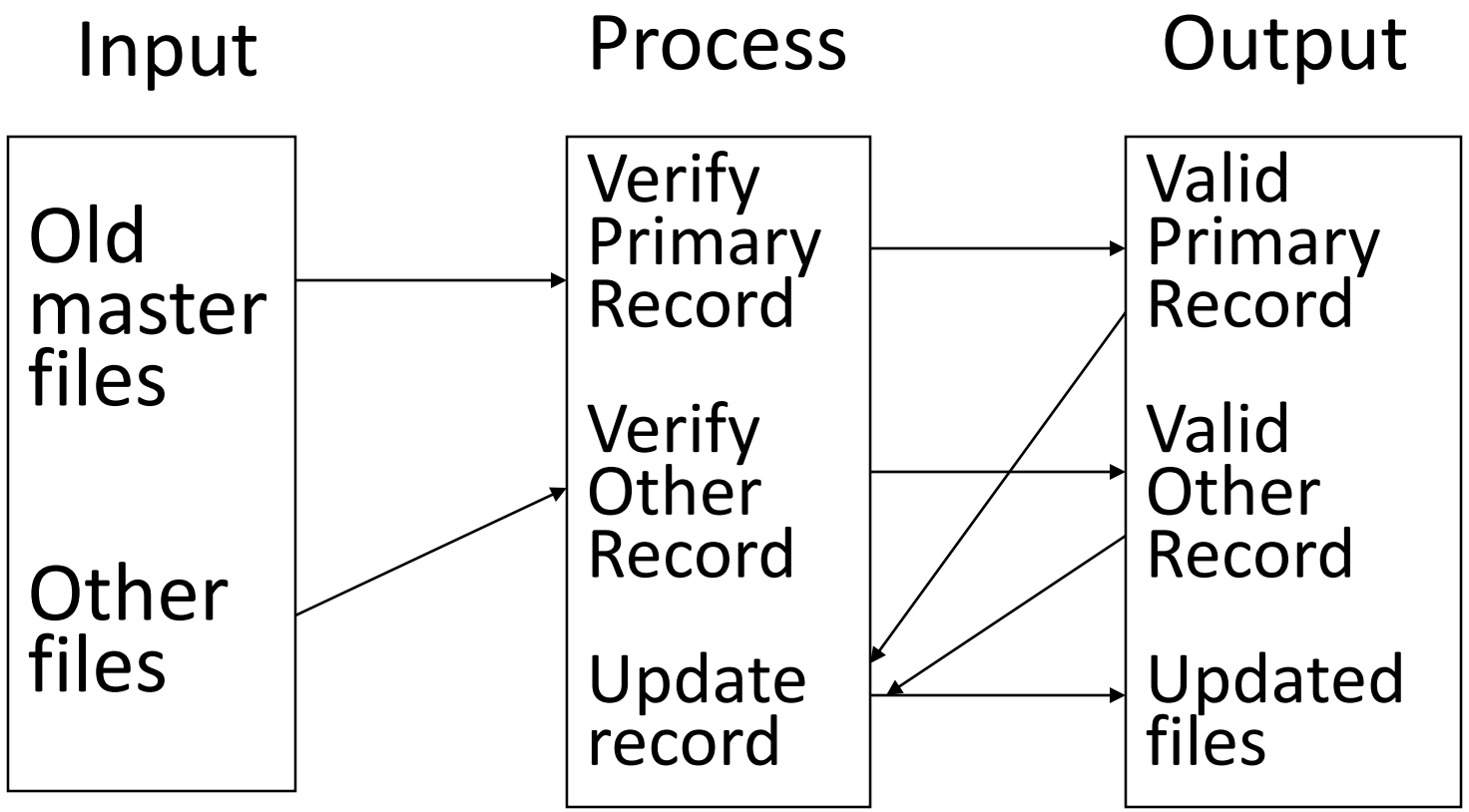
Job *consists of* Task A **and** Task B at the same time; when bother are complete do Task C.

Example of Representing Condition on a Warnier-Orr Diagram

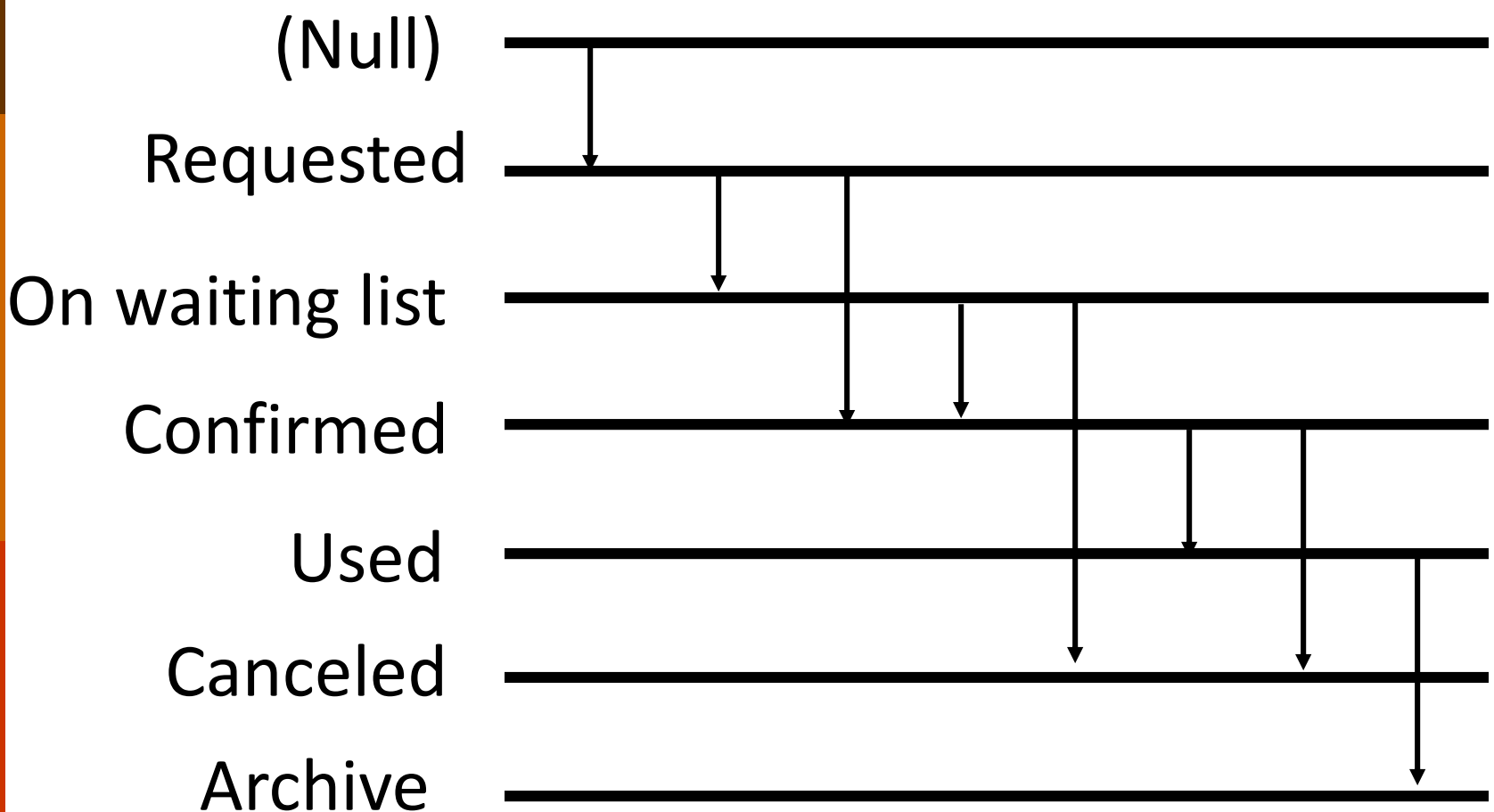


北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

IPO Diagrams



Fence Diagram showing State Transitions (Example: Hotel reservations)



Use UML to Represent OO

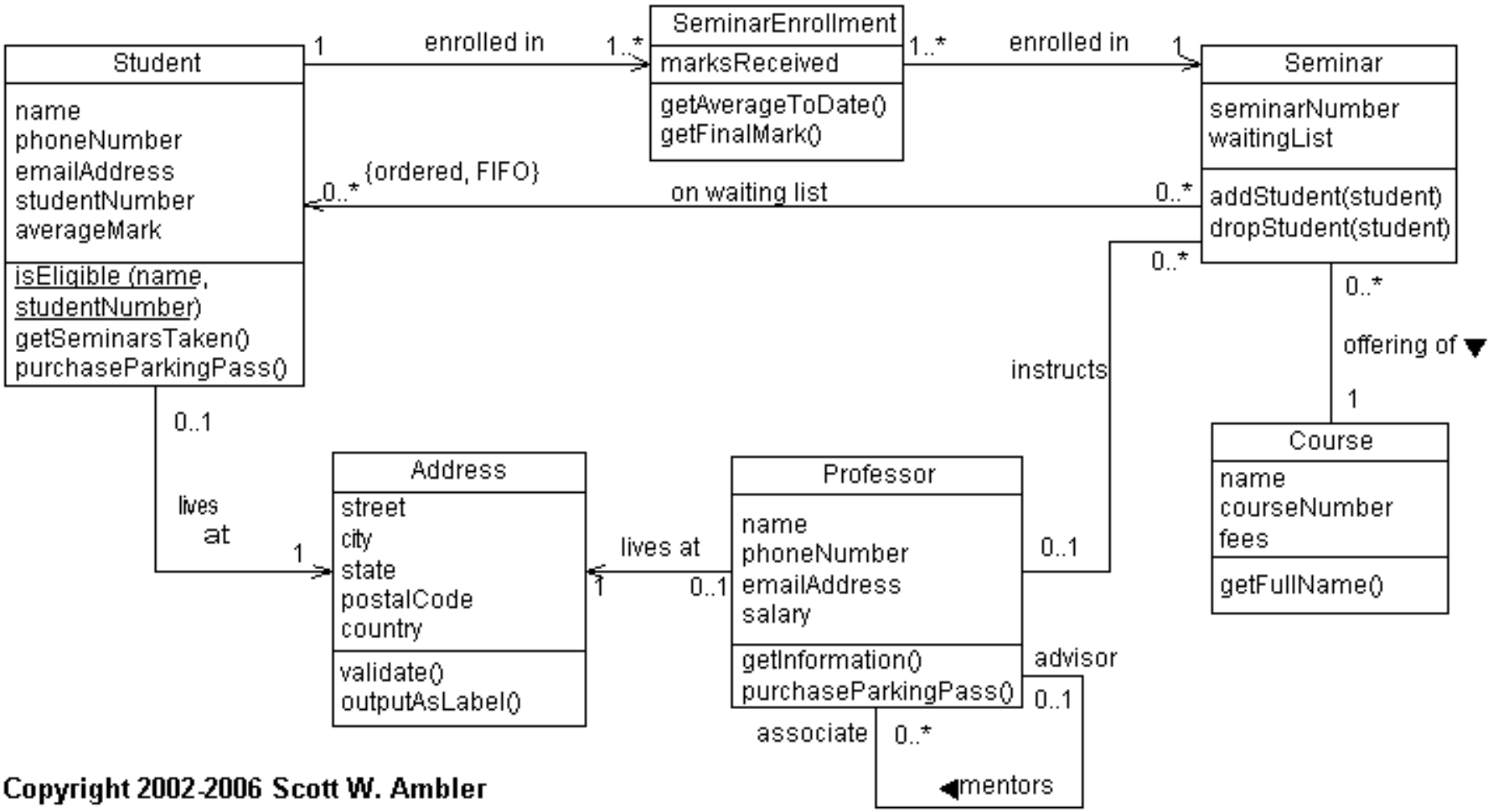
- OMG (Object Management Group) have adopted UML as the OO notational standard.
- UML can be used to visualize, specify, or document a problem.
- UML can be used throughout the software development process.

13 (!!) Kinds of UML Diagrams

1) Activity
2) Class
3) Communication
4) Component
5) Component structure
6) Deployment
7) Interaction
8) Object
9) Package
10) Sequence
11) State machine
12) Timing
13) Use case

北京理工大學 BEIJING INSTITUTE OF TECHNOLOGY

Example: Class Diagram



http://www.agiledata.org/images/oo101ClassDiagram.gif

Class

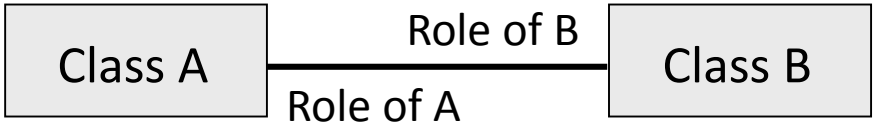
Class Name
Attribute : type
Operation (arg list) : return type *Abstract operation*

Object

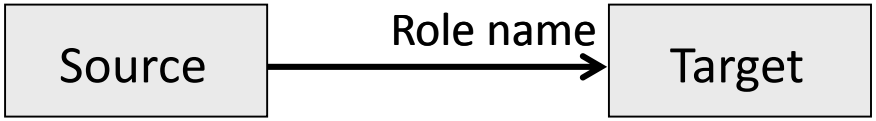
ObjectName: Class Name
Attribute : type
Operation (arg list) : return type *Abstract operation*

Edges

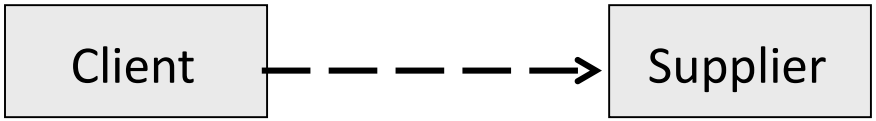
Association



Navigability



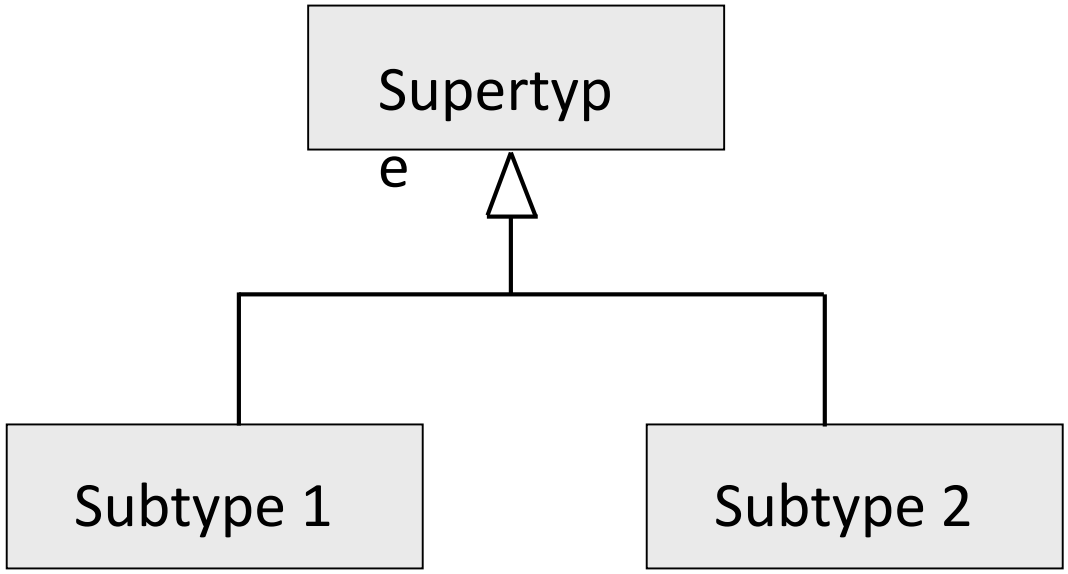
Dependency



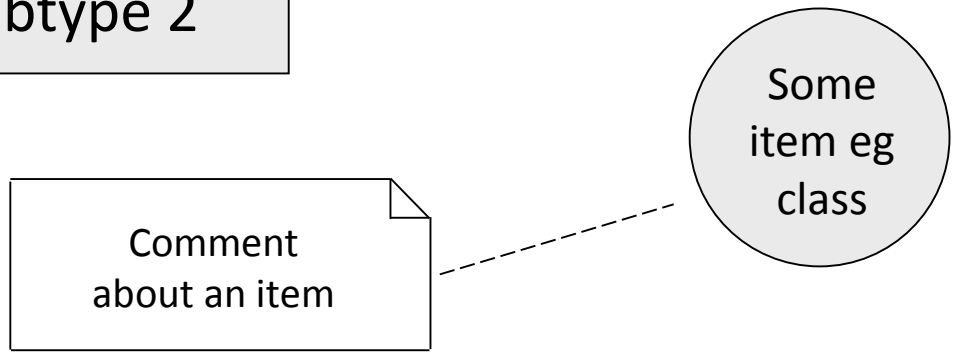
Multiplicities on Edges (Cardinalities)

1	Exactly one
*	Many (any number)
0..1	Optional (zero or one)
m..n	Specified range
{ordered}*	Ordered

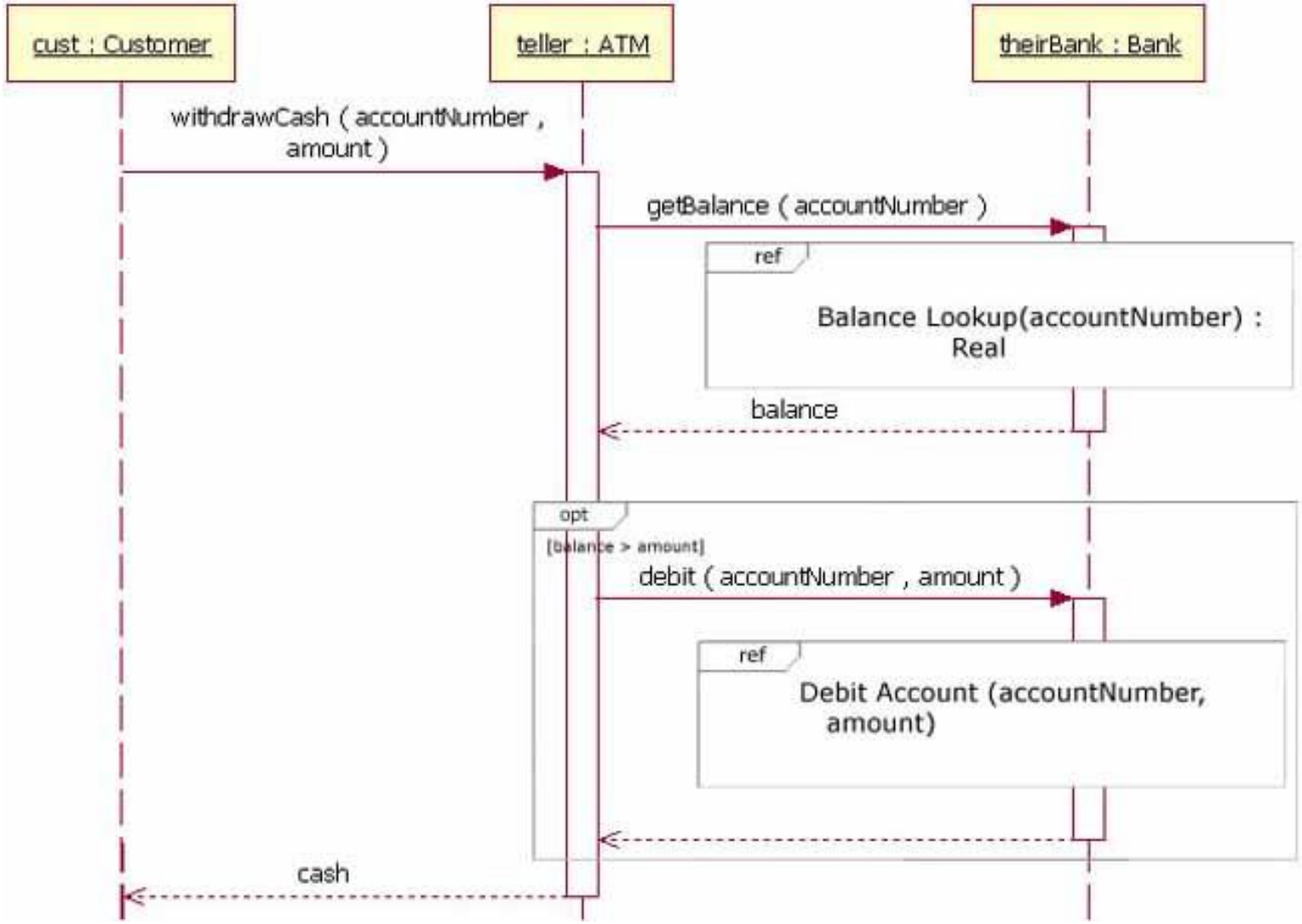
Generalization (Inheritance)



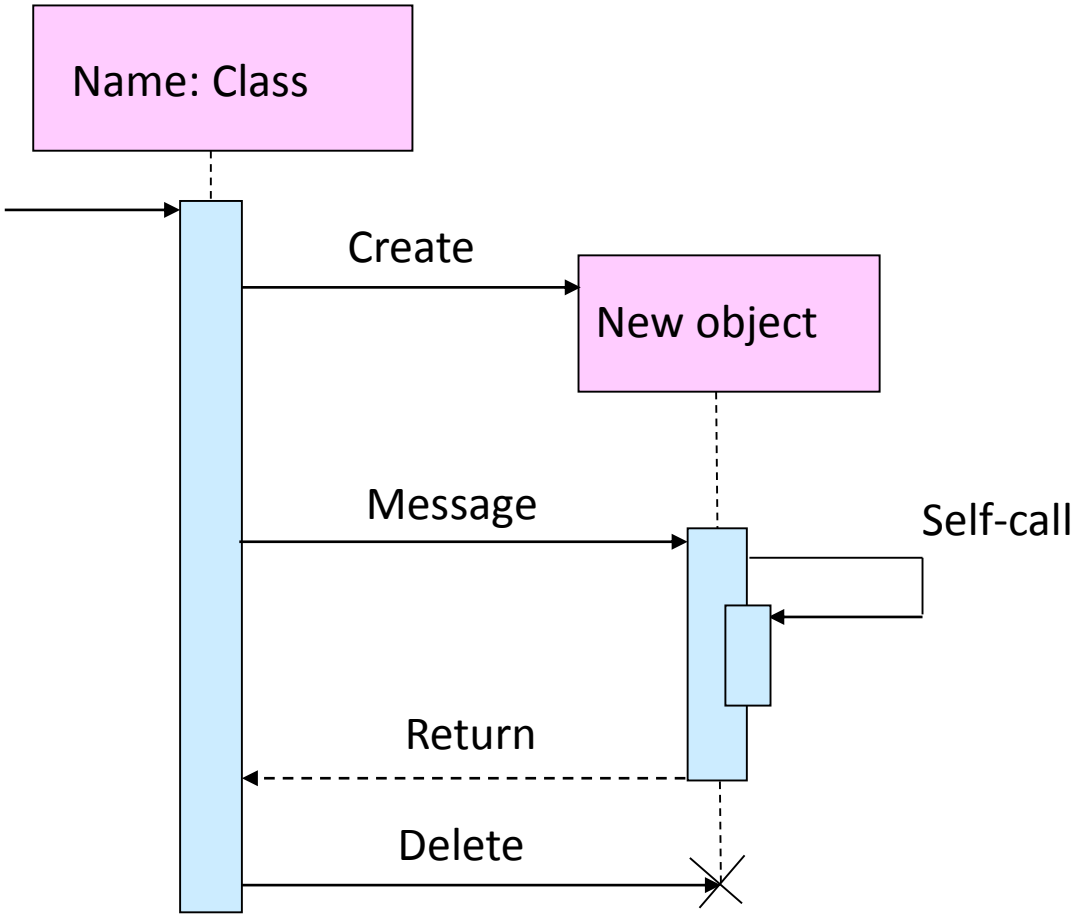
Note (Comment)



Example: Sequence Diagram



Elements of Sequence Diagrams



There is also notation for loops, conditions, etc.

北京理工大學 BEIJING INSTITUTE OF TECHNOLOGY

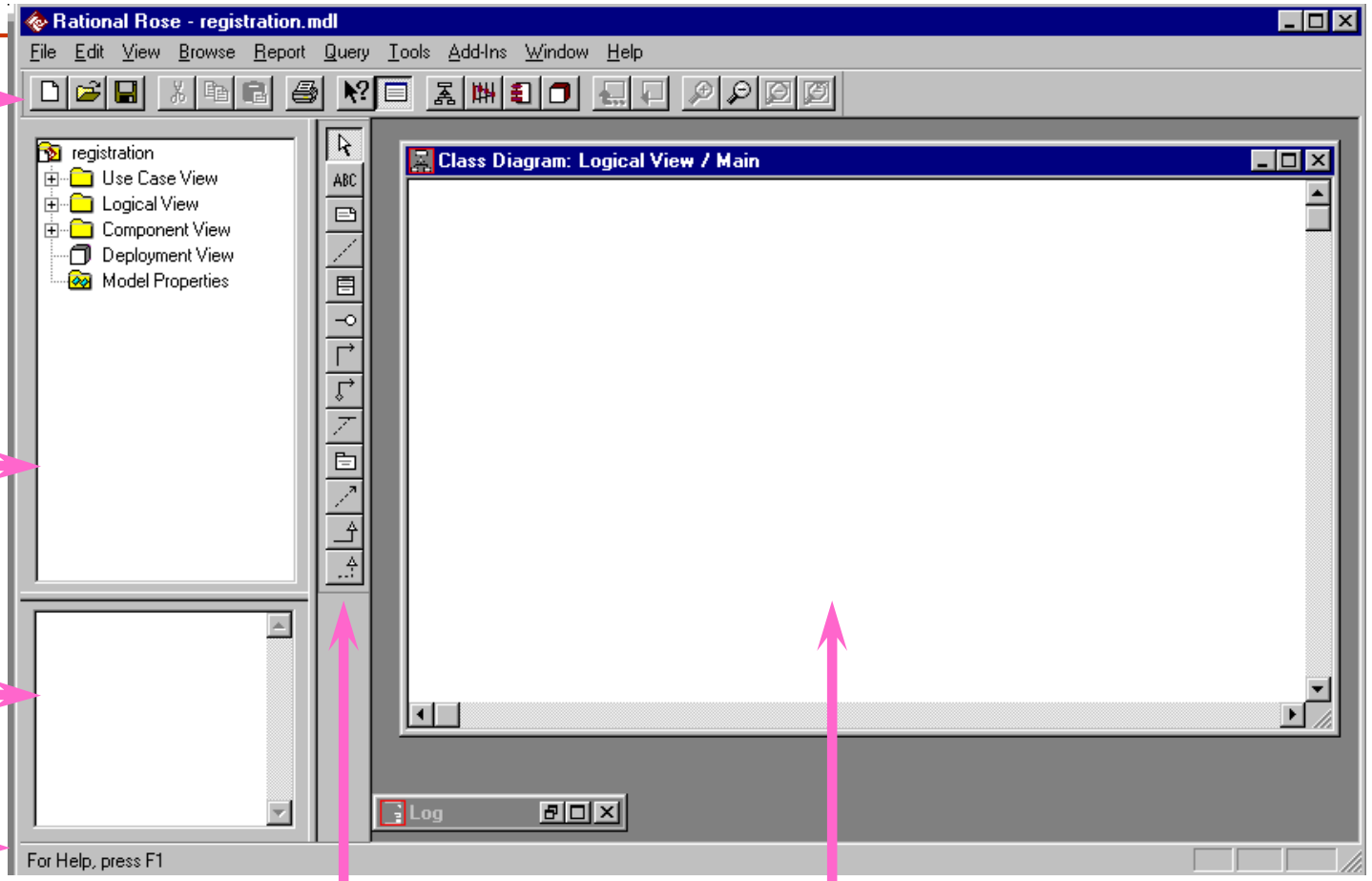
UML Modeling Tools

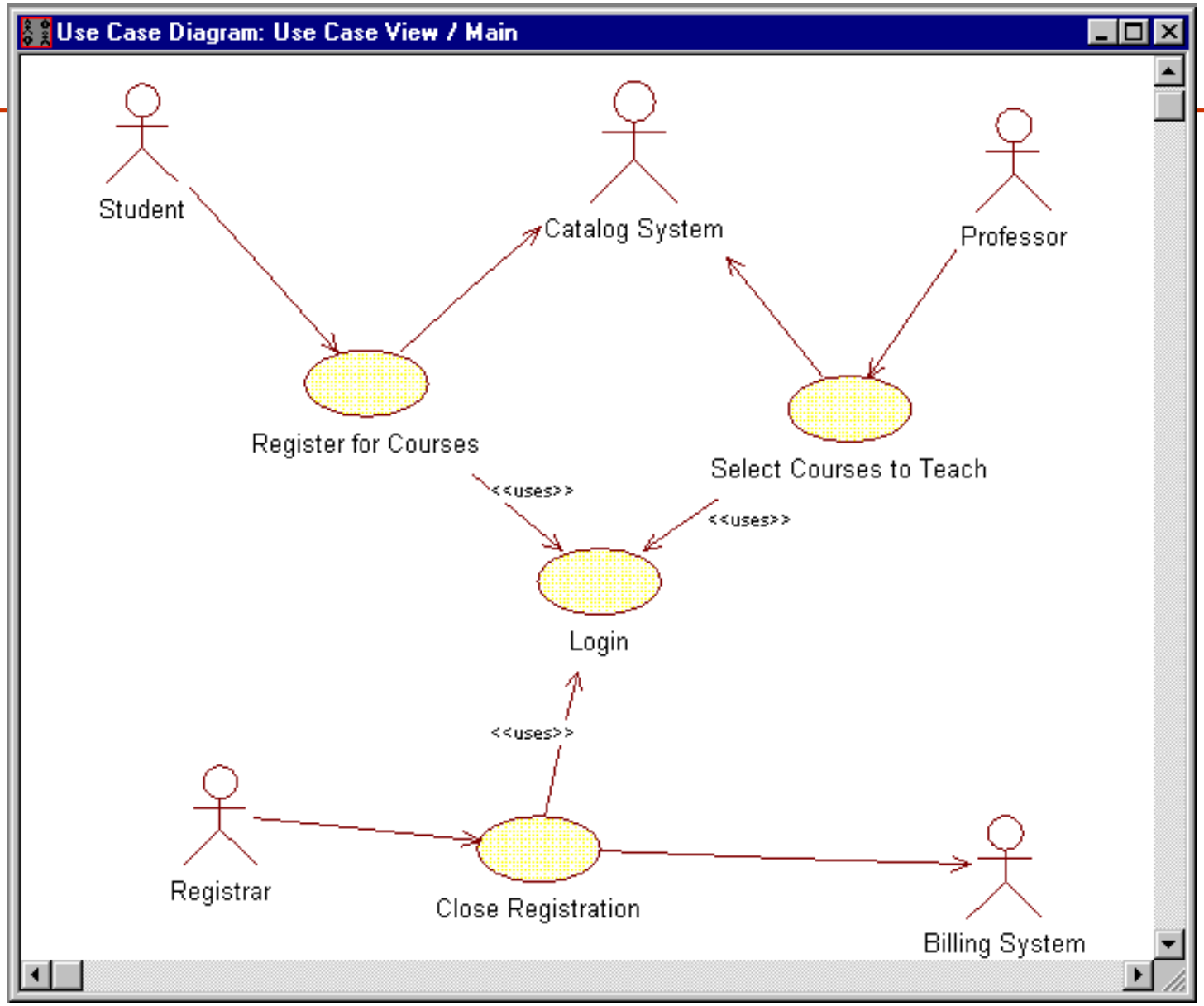
- Rational Rose (www.rational.com) by IBM
- TogetherSoft Control Center, Borland (http://www.borland.com/together/index.html)
- **ArgoUML** (free software) (http://argouml.tigris.org/)
  OpenSource; written in Java
- Others
  http://www.objectsbydesign.com/tools/umltools_byCompany.html

Rational
unifying software teams

北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

Rational Rose




Use Case Diagram: Use Case View / Main
Student
Catalog System
Professor
Register for Courses
Select Courses to Teach
<<uses>>
<<uses>>
Login
<<uses>>
Registrar
Close Registration
Billing System